Star height of regular languages

Thomas Lang

14 July 2014

Albert-Ludwigs-Universität Freiburg

Overview

UNI FREIBURG

Overview

Motivation

Motivation

- Let L_1, L_2 be regular languages.

Motivation

- Let L_1, L_2 be regular languages.
- Aim: Give meaning to the statement

$$L_1 \text{ is more complicated than } L_2.$$

Motivation

- Let L_1, L_2 be regular languages.
- Aim: Give meaning to the statement

$$L_1 \text{ is more complicated than } L_2.$$

- Attempt 1: L_1 is more complicated than L_2 if $|L_1| > |L_2|$

Motivation

- Let L_1, L_2 be regular languages.
- Aim: Give meaning to the statement

$$L_1 \text{ is more complicated than } L_2.$$

- Attempt 1: L_1 is more complicated than L_2 if $|L_1| > |L_2|$
  - Problem: Infinite languages not comparable

UNI FREIBURG

Motivation

- Let L_1, L_2 be regular languages.
- Aim: Give meaning to the statement

$$L_1 \text{ is more complicated than } L_2.$$

- Attempt 1: L_1 is more complicated than L_2 if $|L_1| > |L_2|$
  - Problem: Infinite languages not comparable
- Attempt 2: L_1 is more complicated than L_2 if the minimal automaton of L_1 has more states than the minimal automaton of L_2

Motivation

- Let L_1, L_2 be regular languages.
- Aim: Give meaning to the statement

$$L_1 \text{ is more complicated than } L_2.$$

- Attempt 1: L_1 is more complicated than L_2 if $|L_1| > |L_2|$
  - Problem: Infinite languages not comparable
- Attempt 2: L_1 is more complicated than L_2 if the minimal automaton of L_1 has more states than the minimal automaton of L_2
  - Problem: $\{a^{1000}\}$ more complicated than $\{a^n | n \in \mathbb{N}_0\}$

UNI FREIBURG

Overview

Regular expressions

Let A be an alphabet, then we have:

Regular expressions

Let A be an alphabet, then we have:

- $\varnothing$, ε and $a \in A$ are regular expressions.

UNI FREIBURG

Regular expressions

Let A be an alphabet, then we have:

- $\varnothing$, ε and $a \in A$ are regular expressions.
- If e and e' are regular expressions, then

Regular expressions

Let A be an alphabet, then we have:

- $\varnothing$, ε and $a \in A$ are regular expressions.

- If e and e' are regular expressions, then
  - $(e + e')$,

Regular expressions

Let A be an alphabet, then we have:

- $\varnothing$, ε and $a \in A$ are regular expressions.
- If e and e' are regular expressions, then
  - $(e + e')$,
  - $(e \cdot e')$,

Regular expressions

Let A be an alphabet, then we have:

- $\varnothing$, ε and $a \in A$ are regular expressions.
- If e and e' are regular expressions, then
  - $(e + e')$,
  - $(e \cdot e')$,
  - e^*

  are regular expressions.

Regular expressions

Let A be an alphabet, then we have:

- $\varnothing$, ε and $a \in A$ are regular expressions.
- If e and e' are regular expressions, then
  - $(e + e')$,
  - $(e \cdot e')$,
  - e^*

  are regular expressions.

The language described by a regular expression e is denoted by $\mathcal{L}(e)$.

Star height of regular expressions

Star height of regular expressions

Let e be a regular expression over an alphabet A, then its star height is defined as

Star height of regular expressions

Let e be a regular expression over an alphabet A, then its star height is defined as

- If $e = \emptyset$, $e = \varepsilon$ or $e = a \in A$, then

Star height of regular expressions

Let e be a regular expression over an alphabet A, then its star height is defined as

- If $e = \varnothing$, $e = \varepsilon$ or $e = a \in A$, then

$$\mathrm{h}(e) := 0.$$

Star height of regular expressions

Let e be a regular expression over an alphabet A, then its star height is defined as

- If $e = \varnothing$, $e = \varepsilon$ or $e = a \in A$, then

$$\mathrm{h}(e) := 0.$$

- If $e = e' + e''$ or $e = e' \cdot e''$, then

Star height of regular expressions

Let e be a regular expression over an alphabet A, then its star height is defined as

- If $e = \varnothing$, $e = \varepsilon$ or $e = a \in A$, then
$$\mathrm{h}(e) := 0.$$
- If $e = e' + e''$ or $e = e' \cdot e''$, then
$$\mathrm{h}(e) := \max(h(e'), h(e'')).$$

Star height of regular expressions

Let e be a regular expression over an alphabet A, then its star height is defined as

- If $e = \emptyset$, $e = \varepsilon$ or $e = a \in A$, then

$$\mathrm{h}(e) := 0.$$

- If $e = e' + e''$ or $e = e' \cdot e''$, then

$$\mathrm{h}(e) := \max(h(e'), h(e'')).$$

- If $e = e'^*$, then

Star height of regular expressions

Let e be a regular expression over an alphabet A, then its star height is defined as

- If $e = \varnothing$, $e = \varepsilon$ or $e = a \in A$, then
$$\mathrm{h}(e) := 0.$$
- If $e = e' + e''$ or $e = e' \cdot e''$, then
$$\mathrm{h}(e) := \max(h(e'), h(e'')).$$
- If $e = e'^*$, then
$$\mathrm{h}(e) := 1 + h(e').$$

Examples

Examples

- $e_1 := a^*(ba^*)^*$ $\Rightarrow$ $\mathrm{h}(e_1) =$

Examples

- $e_1 := a^*(ba^*)^*$ $\Rightarrow$ $\mathrm{h}(e_1) = 2$

Examples

- $e_1 := a^*(ba^*)^* \quad \Rightarrow \quad \mathrm{h}(e_1) = 2$
- $e_2 := a^* + ((b^*ab^*)^*a)^* \quad \Rightarrow \quad \mathrm{h}(e_2) =$

Examples

- $e_1 := a^*(ba^*)^* \quad \Rightarrow \quad \mathrm{h}(e_1) = 2$
- $e_2 := a^* + ((b^*ab^*)^*a)^* \quad \Rightarrow \quad \mathrm{h}(e_2) = 3$

Examples

- $e_1 := a^*(ba^*)^* \quad \Rightarrow \quad \mathrm{h}(e_1) = 2$
- $e_2 := a^* + ((b^*ab^*)^*a)^* \quad \Rightarrow \quad \mathrm{h}(e_2) = 3$
- Caution: $\mathcal{L}(a^*) = \mathcal{L}((a^*)^*)$, but

UNI FREIBURG

Examples

- $e_1 := a^*(ba^*)^* \quad \Rightarrow \quad \mathrm{h}(e_1) = 2$
- $e_2 := a^* + ((b^*ab^*)^*a)^* \quad \Rightarrow \quad \mathrm{h}(e_2) = 3$
- Caution: $\mathcal{L}(a^*) = \mathcal{L}((a^*)^*)$, but $\mathrm{h}(a^*) = 1 \neq 2 = \mathrm{h}((a^*)^*)$

Star height of regular languages

Star height of regular languages

Let L be a regular language, then its star height is defined as

Star height of regular languages

Let L be a regular language, then its star height is defined as

$$\mathrm{h}(L) := \min(\{\mathrm{h}(e) \mid \mathcal{L}(e) = L\}).$$

Overview

Description

Description

- Brzozowski-McCluskey algorithm

Description

- Brzozowski-McCluskey algorithm
- Operates on generalized automaton $\mathcal{A}$, i.e. an automaton whose edges are labeled with regular expressions

Description

- Brzozowski-McCluskey algorithm
- Operates on generalized automaton $\mathcal{A}$, i.e. an automaton whose edges are labeled with regular expressions
- Computes regular expression e with $\mathcal{L}(e) = \mathcal{L}(\mathcal{A})$

BMC algorithm

BMC algorithm

1. Insert new state i and ε-transitions from i to all initial states

BMC algorithm

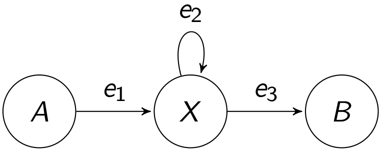
1. Insert new state i and ε-transitions from i to all initial states
2. Insert new state t and ε-transitions from all final states to t

BMC algorithm

1. Insert new state i and ε-transitions from i to all initial states
2. Insert new state t and ε-transitions from all final states to t
3. Successively remove the states of $\mathcal{A}$, updating the edges in the following manner:

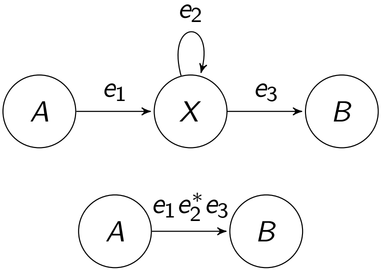
BMC algorithm

1. Insert new state i and ε-transitions from i to all initial states
2. Insert new state t and ε-transitions from all final states to t
3. Successively remove the states of $\mathcal{A}$, updating the edges in the following manner:



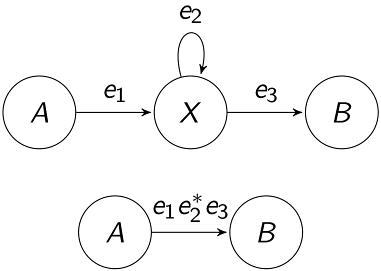
BMC algorithm

1. Insert new state i and ε-transitions from i to all initial states
2. Insert new state t and ε-transitions from all final states to t
3. Successively remove the states of $\mathcal{A}$, updating the edges in the following manner:



BMC algorithm

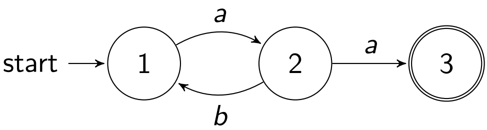
1. Insert new state i and ε-transitions from i to all initial states
2. Insert new state t and ε-transitions from all final states to t
3. Successively remove the states of $\mathcal{A}$, updating the edges in the following manner:



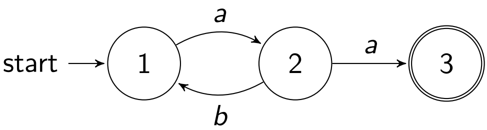
4. Join all expressions on edges from i to t using '$+$'

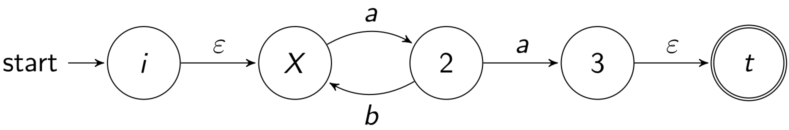
Example

Example

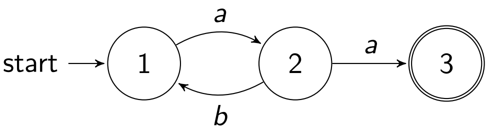


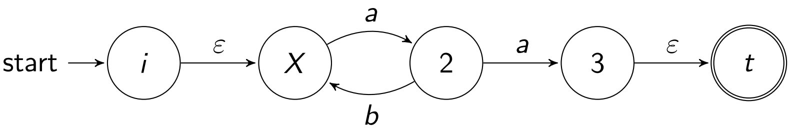
Example

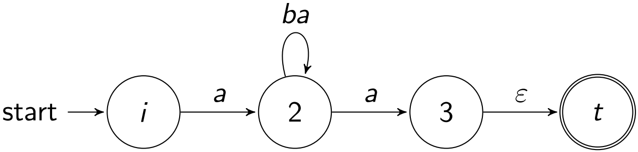




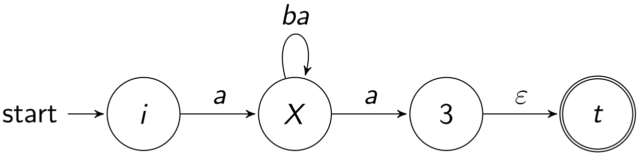
Example



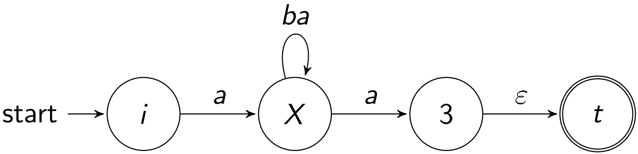


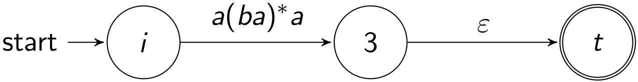


Example continued

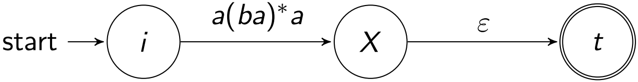


Example continued

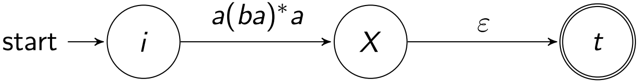


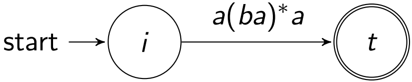


Example continued



Example continued





Overview

Graph-theoretic concepts

Graph-theoretic concepts

A set of vertices V of a graph is called

Graph-theoretic concepts

A set of vertices V of a graph is called

- *strongly connected*, if every vertex in V is reachable by every other vertex in V.

Graph-theoretic concepts

A set of vertices V of a graph is called

- *strongly connected*, if every vertex in V is reachable by every other vertex in V.
- a *ball*, if V is strongly connected and has at least one edge.

Examples

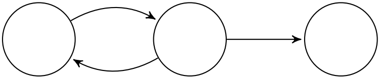


Figure : A graph

Examples

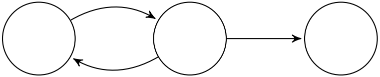


Figure : A graph

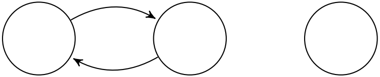


Figure : Its strongly connected components

Examples

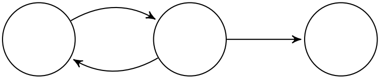


Figure : A graph

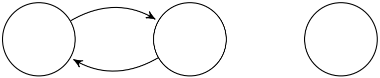


Figure : Its strongly connected components

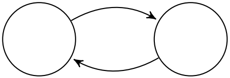


Figure : Its ball

Loop complexity

Loop complexity

Let G be a graph, then its loop complexity is defined as follows:

Loop complexity

Let G be a graph, then its loop complexity is defined as follows:

- If G contains no ball

Loop complexity

Let G be a graph, then its loop complexity is defined as follows:

- If G contains no ball

$$\mathrm{lc}(G) := 0$$

Loop complexity

Let G be a graph, then its loop complexity is defined as follows:

- If G contains no ball

$$\mathrm{lc}(G) := 0$$

- If G is not a ball

Loop complexity

Let G be a graph, then its loop complexity is defined as follows:

- If G contains no ball

$$\mathsf{lc}(G) := 0$$

- If G is not a ball

$$\mathsf{lc}(G) := \max(\{\mathsf{lc}(B) | B \text{ is a ball of } G\})$$

Loop complexity

Let G be a graph, then its loop complexity is defined as follows:

- If G contains no ball

$$\mathsf{lc}(G) := 0$$

- If G is not a ball

$$\mathsf{lc}(G) := \max(\{\mathsf{lc}(B) | B \text{ is a ball of } G\})$$

- If G is a ball

Loop complexity

Let G be a graph, then its loop complexity is defined as follows:

- If G contains no ball

$$\mathsf{lc}(G) := 0$$

- If G is not a ball

$$\mathsf{lc}(G) := \max(\{\mathsf{lc}(B) | B \text{ is a ball of } G\})$$

- If G is a ball

$$\mathsf{lc}(G) := 1 + \min(\{\mathsf{lc}(G \setminus \{v\}) | v \text{ is a vertex of } G\})$$

Examples

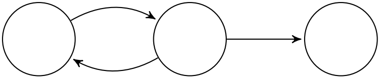


Figure : A graph G

Examples

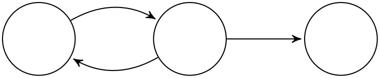


Figure : A graph G

We have $\mathrm{lc}(G) = 1$.

Overview

UNI FREIBURG

Theorem

The loop complexity of a trim automaton $\mathcal{A}$ is equal to the minimum of the star heights of the expressions obtained by the different possible runs of the BMC algorithm on $\mathcal{A}$.

Theorem

The loop complexity of a trim automaton $\mathcal{A}$ is equal to the minimum of the star heights of the expressions obtained by the different possible runs of the BMC algorithm on $\mathcal{A}$.

In general, given an automaton $\mathcal{A}$ and two total orders on its states ω_1 and ω_2, we have

Theorem

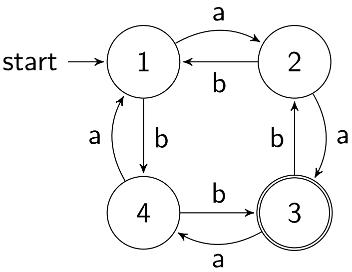
The loop complexity of a trim automaton $\mathcal{A}$ is equal to the minimum of the star heights of the expressions obtained by the different possible runs of the BMC algorithm on $\mathcal{A}$.

In general, given an automaton $\mathcal{A}$ and two total orders on its states ω_1 and ω_2, we have

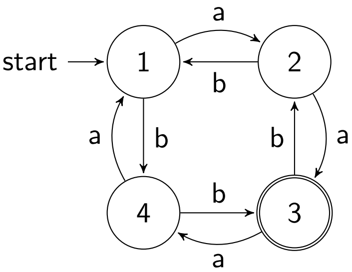
$$\mathsf{h}(\mathsf{BMC}(\mathcal{A}, \omega_1)) \neq \mathsf{h}(\mathsf{BMC}(\mathcal{A}, \omega_2))$$

as the following example shows:

Example

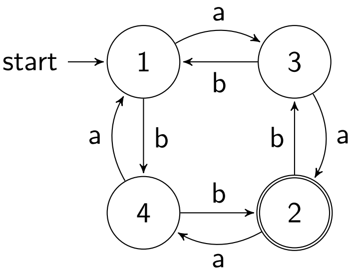


Example

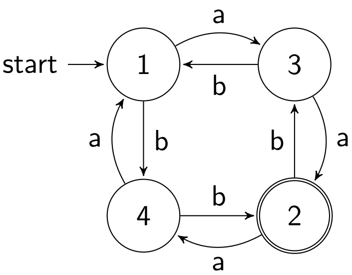


Star height of result of BMC algorithm: 3

Example continued



Example continued



Star height of result of BMC algorithm: 2

Minimal automata

Minimal automata

- Consider now a regular language L.

Minimal automata

- Consider now a regular language L.
- Does the loop complexity of its minimal automaton correspond to $\mathrm{h}(L)$?

Minimal automata

- Consider now a regular language L.
- Does the loop complexity of its minimal automaton correspond to $\mathrm{h}(L)$?
- Not necessarily, as the following example shows.

Example

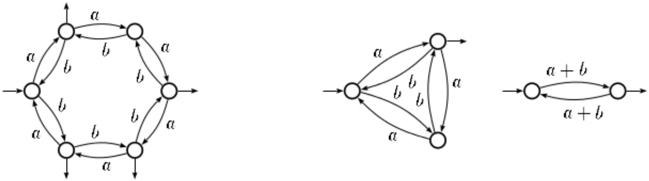


Figure : Minimal automata $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ for their respective languages[1]

[1] LoSa00, On the star height of regular languages

Example

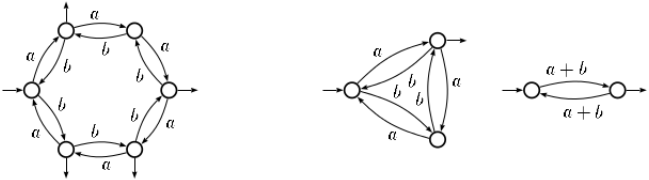


Figure : Minimal automata $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ for their respective languages[1]

- $\mathrm{lc}(\mathcal{A}) = 3$,

[1]LoSa00, On the star height of regular languages

Example

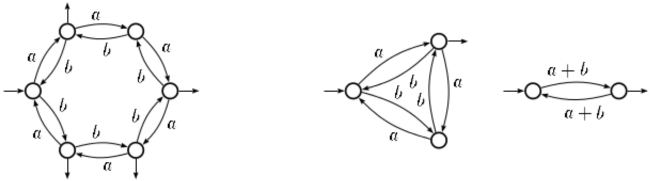


Figure : Minimal automata $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ for their respective languages[1]

- $\mathrm{lc}(\mathcal{A}) = 3$, $\mathrm{lc}(\mathcal{B}) = 2$,

[1] LoSa00, On the star height of regular languages

Example

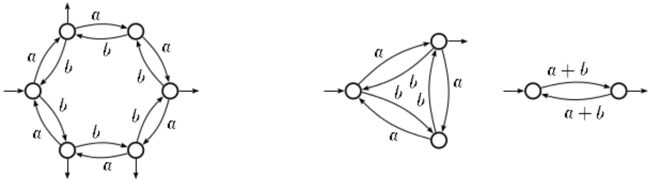


Figure : Minimal automata $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ for their respective languages[1]

- $\mathrm{lc}(\mathcal{A}) = 3$, $\mathrm{lc}(\mathcal{B}) = 2$, $\mathrm{lc}(\mathcal{C}) = 1$

[1]LoSa00, On the star height of regular languages

Example

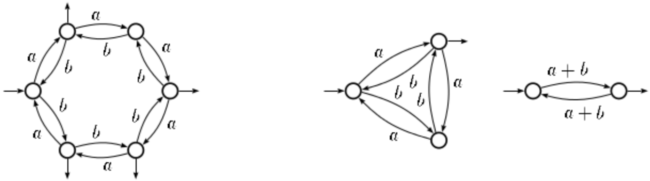


Figure : Minimal automata $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ for their respective languages[1]

- $\mathrm{lc}(\mathcal{A}) = 3$, $\mathrm{lc}(\mathcal{B}) = 2$, $\mathrm{lc}(\mathcal{C}) = 1$
- But: $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B}) \cup \mathcal{L}(\mathcal{C})$

[1] LoSa00, On the star height of regular languages

Example

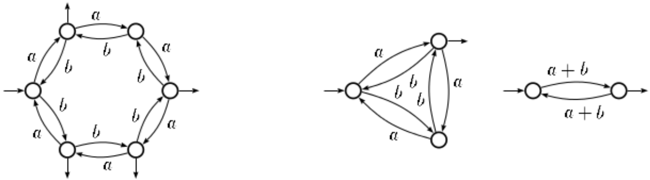


Figure : Minimal automata $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ for their respective languages[1]

- $\mathrm{lc}(\mathcal{A}) = 3$, $\mathrm{lc}(\mathcal{B}) = 2$, $\mathrm{lc}(\mathcal{C}) = 1$
- But: $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{B}) \cup \mathcal{L}(\mathcal{C})$
- Therefore: $\mathrm{h}(\mathcal{L}(\mathcal{A})) \leq 2 < 3 = \mathrm{lc}(\mathcal{A})$

[1] LoSa00, On the star height of regular languages

Computability

Computability

- Is the star height of a regular language computable?

Computability

- Is the star height of a regular language computable?
- Not in general

Computability

- Is the star height of a regular language computable?
- Not in general
- But: For a subset of the regular languages (*pure-group languages*) it is computable

Overview

References

[LoSa00] Lombardy, S.; Sakarovitch, J.: On the star height of rational languages. In: *Proceedings of the 3rd International Conference on Words, Languages and Combinatorics, Kyoto (Japan), March 2000*